시스템 설계 문서 (SDD)

프로젝트 제목: 자율 이동 로봇(AMR) 순찰 및 보안 시스템 – 진돗봇
버전: 1.1
날짜: 2025.05.31

---

1. 개요

자율 이동 로봇(AMR) 보안 시스템은 두 대의 AI 기반 TurtleBot4 로봇(TurtleBot4A, TurtleBot4B)을 사용하여 보안 구역 내에서 자율 순찰, 화재 탐지 및 진압, 직원 대피 유도, 침입자 탐색 및 추적을 제공하도록 설계되었습니다. 시스템은 두 대의 AMR이 독립적으로 작동하되, 특정 시나리오에서 협력하며 데이터를 현장에서 처리합니다.

---

2. 배경

[키워드PICK] 반복되는 '물류센터 화재'...진정한 해결책은 무엇인가

김재황 기자 eltred@hellot.net | 등록 2025.05.21 21:20:07 URL복사

▲ 지난 2021년 발생한 쿠팡 물류센터 화재 사고 후 모습 / 출처 - 연합뉴스

경기도 이천에서 다시 한 번 물류센터 화재가 발생했다. 지난 13일 오전, 냉동식품과 생활용품, 리튬이온 배터리 등 다양한 품목이 보관된 대형 물류창고에서 화염이 치솟았다. 정확한 화재 원인은 공식 발표를 기다리고 있지만 한동안 잠잠했던 물류센터 화재 이슈가 재부상하면서 업계와 사회에 다시금 경고음을 울리고 있다. 언제든 폭발할 수 있는 가능성이 상존하는 휴화산처럼 물류센터 화재 사고는 시대를 가리지 않고 반복되고 있다. 이 고리를 끊기 위한 방법은 무엇일까?

-

출처:https://www.hellot.net/news/article.html?no=101528

현대의 물류센터는 리튬 이온 배터리, 전자기기, 멀티탭 등 화재 위험 요소가 밀집된 구조를 가지고 있으며, 완전 자동화에 가까워질수록 야간 무인 환경이 많아져 사고 발생 시 초기 대응의 어려움이 더욱 부각되고 있습니다. 특히 한 번의 화재가 수십억 원 규모의 피해로 이어지는 만큼, 사전 탐지와 초기 진압은 무엇보다 중요합니다.

또한 물류센터나 자동화 공장과 같은 대규모 실내 공간에서는 외부 침입자의 접근도 또 다른 위협 요소입니다. 기존 경비 시스템은 주로 모니터링 및 알림 수준에 머물러 있으며, 실제 침입 상황 발생 시에는 즉각적인 물리적 대응이 불가능합니다.

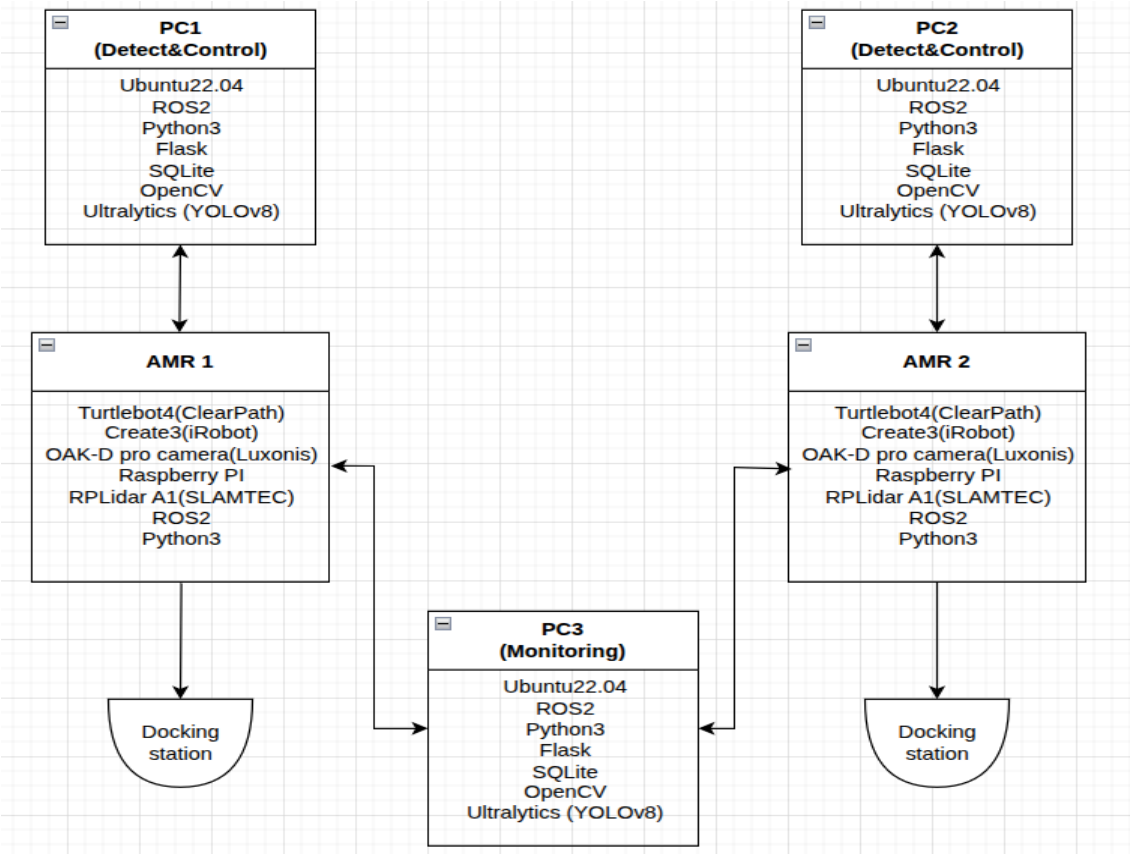
이러한 배경 속에서, 본 프로젝트는 단순한 감시·탐지 수준을 넘어서 화재와 침입 상황에 실제 물리적으로 대응할 수 있는 자율주행형 로봇 시스템을 개발하고자 합니다.

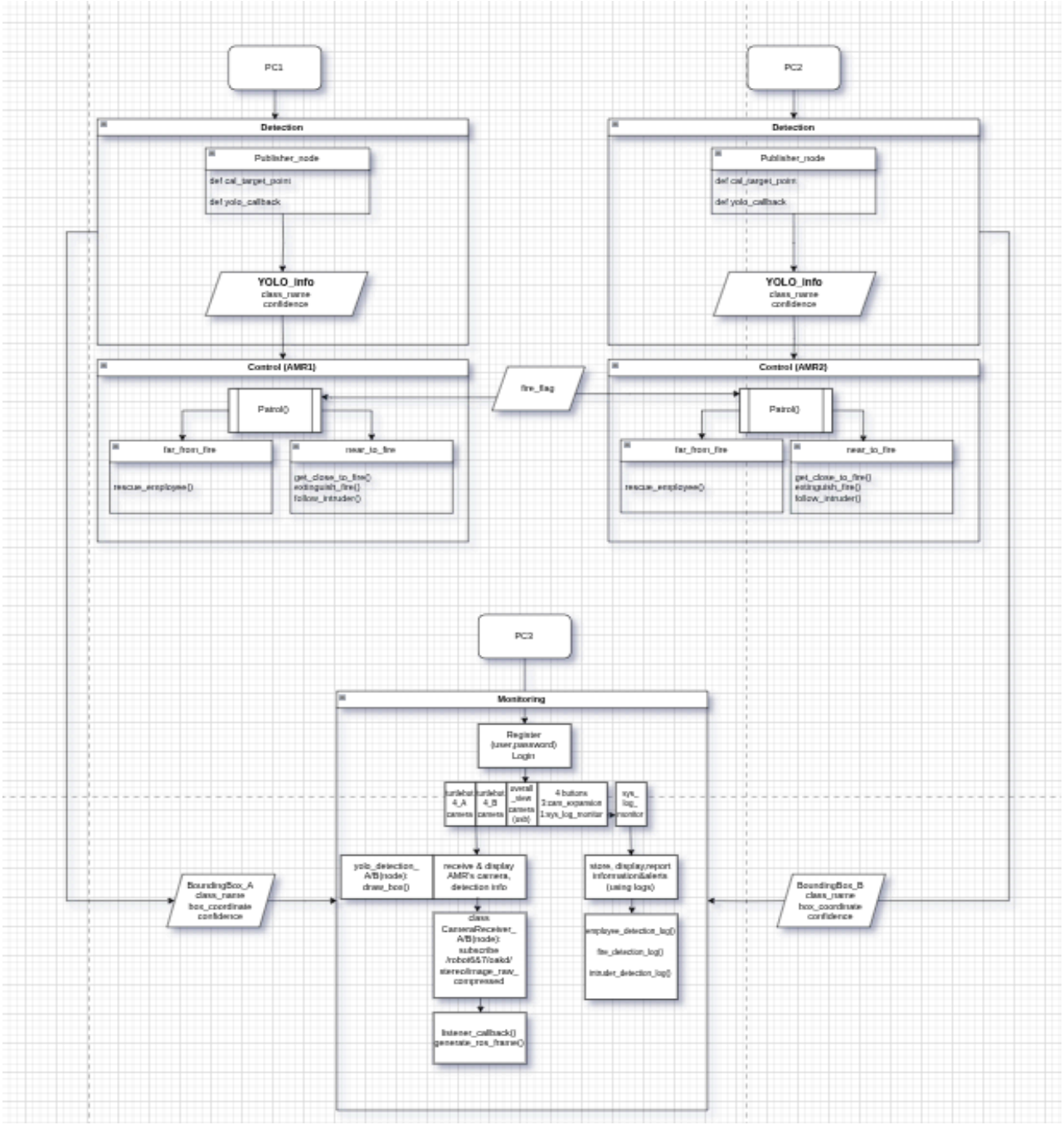
---

3. 시스템 아키텍처

이 시스템은 두 대의 AMR(TurtleBot4A, TurtleBot4B)로 구성되며, 데이터 처리, 네비게이션, 화재 탐지, 위협 탐지, 대피 유도 및 경고가 모두 AMR에서 로컬로 수행됩니다. AMR은 모니터링, 알림 및 수동 제어를 위해 PC의 사용자 인터페이스와 로컬 네트워크(Wi-Fi)를 통해 직접 통신합니다.

3.1 고레벨 아키텍처 다이어그램





---

4. 구성 요소 설계

4.1 하드웨어 구성 요소

1. **PC (모니터링 시스템)**
    - **운영 체제:** Ubuntu 22.04
    - **카메라:** 로컬 시각적 모니터링을 위한 USB 카메라 (필요한 경우)
    - **네트워크:** AMR과 안정적인 Wi-Fi 6 연결 지원

2. **자율 이동 로봇 (AMR) – TurtleBot4**
    - **프로세서:** Raspberry Pi 4 (온보드 AI 처리 및 데이터 처리)
    - **운영 체제:** Ubuntu 22.04 및 ROS2 (로봇 제어)
    - **센서:**
        - **LiDAR:** 네비게이션 및 장애물 감지
        - **USB 카메라:** 위협 탐지용 비디오 피드 캡처
        - **Depth Camera:** 공간 인식, 사람/장애물 거리 측정
    - **배터리:** 최소 8시간 배터리 수명, 자율 도킹 및 충전 지원
    - **추가 장비:**
        - 소형 소화 장치 : 화재 발생 시 대응용 (투척형 또는 소형 분사형)
        - 테이저 : 침입자 무력화 (비살상 제압 장비)
        - LED 경고등: 비상 시 시각적 알림 제공 (경고 색상 구분)
        - 음성 경고 시스템 (TTS스피커) : 상황 발생 시 실시간 음성으로 경고 알림
3. **충전 스테이션**
    - 수동 개입 없이 AMR의 충전을 위한 자율 도킹 스테이션

4.2 소프트웨어 구성 요소

1. **PC 소프트웨어**
    - **Python3:** 인터페이스 관련 작업을 위한 기본 언어
    - **ROS2:** AMR과의 통신을 위한 플랫폼
    - **Flask:** 모니터링 대시보드를 위한 로컬 웹 서버
    - **SQLite3:** 경고 및 로그 기록을 위한 경량 데이터베이스
    - **OpenCV :** 이미지처리
    - **Ultralytics (YOLO):** PC에서 추가 객체 탐지가 필요한 경우의 이미지 처리 용도
2. **AMR 소프트웨어 (TurtleBot4)**
    - **ROS2 (로봇 운영 시스템 2):** 네비게이션, 제어 및 데이터 통합의 핵심 플랫폼
    - **Python3:** AMR 작동 및 AI 처리에 사용하는 기본 프로그래밍 언어
    - **OpenCV 및 Ultralytics (YOLO):** 객체 인식에 사용되며 화재 및 직원 위치, 침입자 추적

---

5. 데이터 흐름 설계

1. **데이터 수집**
    - **AMR 센서:** AMR 2대 (TurtleBot4)가 LiDAR, usb camera, depth camera를 사용하여 자율 네비게이션을 수행하고 잠재적 위협(화재 및 침입자)을 감지하며, 직원들의 위치 좌표를 수집합니다.
2. **데이터 처리 및 분석**
    - **PC(victus) 처리:** AMR에서 수집된 비디오 및 센서 데이터는 PC에서 YOLO를 사용해 로컬로 처리되어 객체 감지 및 이미지 분석이 이루어집니다. 또한 PC에서 AI 추론을 수행하여 실시간으로 객체(직원, 침입자)나 이상 상태(화재)를 식별합니다.
3. **데이터 저장**
    - **PC 로그 저장:** 중요 데이터를 로컬 PC에 48시간 동안 저장하며, 경고 및 로그가 모니터링 PC의 SQLite3에 저장되어 검토할 수 있습니다.
4. **경고 및 알림**
    - **실시간 경고 전송:** 잠재적 위협이 감지되면 AMR은 Wi-Fi를 통해 모니터링 PC로 경고를 전송합니다.
    - **웹 대시보드 표시:** PC 웹 대시보드에서 보안 담당자는 실시간 비디오 피드, 경고 및 AMR 상태를 확인할 수 있습니다.

---

6. 상세 설명

6.1 객체 탐지 기능 : Raspberry-pi에서 실행되는 YOLO 기반 객체 감지

- **화재 탐지:** 화재를 식별합니다.
- **침입자 탐지**: 사람을 식별하고, 일반 직원과 침입자를 구분합니다.
- **인명 탐지**: 연기나 장애물에 가려진 상황에서도 구조 대상자를 식별할 수 있도록 훈련된 객체 감지 모델을 사용합니다.

6.2 AMR 네비게이션 및 기능

- **TurtleBot4: 두 대의 AMR이 각각 역할을 분담하여 작동합니다.**
    - **AMR-1:** 화재 감지, 초기 진압, 침입자 탐지 및 추적 수행
    - **AMR-2:** 인명 구조 전담 – 위험 지역 내에 잔류한 인원을 탐지하고 대피를 유도하거나 구조 요청 전달
- **네비게이션:** ROS2와 SLAM(동시 위치 추정 및 매핑) 및 LiDAR를 사용하여 AMR이 구역 내에서 자율적으로 이동하고 장애물을 대피합니다.
- **화재 진압:** 화재 탐지 시 투척형 소화 장치 사용하여 초기 화재 진압

- **침입자 대응**: 화재 진압 이후, 침입자를 탐지하고 위치 추적하며, 위험 지역 이탈을 유도하거나 경고합니다.
- **인명 구조**: 감지된 인원을 향해 접근하여 대피 경로를 안내하거나, 위험 상황 시 대시보드를 통해 구조 요청을 전송합니다. 장애물 우회 기능을 통해 인접 지역 탐색도 수행합니다.

6.3 사용자 인터페이스 (대시보드)

- **모니터링 대시보드 (Flask 기반):**
    - 두 대의 AMR의 실시간 상태, 라이브 비디오 피드, 경고를 표시합니다.
    - Flask로 구축되었으며 PC의 브라우저에서 접근할 수 있어 담당자가 AMR을 모니터링하고 제어할 수 있습니다.
    - **수동 제어:** 담당자는 필요시 ROS2 명령을 통해 수동으로 AMR을 제어할 수 있습니다.

---

7. 보안 설계

- **데이터 암호화:** AMR과 PC 간 전송되는 모든 데이터는 TLS 1.3을 사용해 암호화됩니다.
- **접근 제어:** 대시보드는 다중 요소 인증을 통해 접근이 제한되어 있습니다.
- **로컬 저장 보안:** PC에 저장된 로그 및 경고 데이터는 AES-256을 사용해 암호화하여 민감한 정보를 보호합니다.

---

8. 성능 요구사항

- **지연 시간:** 경고는 1초 미만의 지연 시간으로 모니터링 PC에 도달해야 합니다.
- **시스템 가동률:** AMR은 최소 8시간 동안 연속 운영이 가능해야 하며, 필요시 충전 독으로 복귀할 수 있어야 합니다.
- **확장성:** 두 대의 AMR 운영에 맞춰 설계되었으나, 추가 유닛이 필요할 경우 최소한의 아키텍처 변경으로 지원 가능합니다.
- **침입자 및 화재 탐지 정확도:** 침입자 및 화재 감지는 95% 이상의 탐지 정확도를 유지해야 합니다.
- **초기 대응 시간:** 이상 상황 발생 후 3초 이내에 제압/소화 장치를 작동시켜야 합니다.
- **동시 운영 처리:** 여러대의 로봇이 동시 운영되어야 하는 만큼 충돌 없는 경로 계획 및 각 로봇이 상황에 맞는 대응을 수행해야 합니다.

---

9. 오류 처리 및 복구

- **네트워크 손실:** Wi-Fi 연결이 끊긴 경우에도 로봇은 자율 주행 및 감지 기능은 유지하되, 로그는 로컬에 임시저장합니다. 연결이 복구되면 로컬에 저장되어있는 누락된 데이터를 전송합니다.
- **하드웨어 오류:** AMR은 주기적으로 센서, 카메라, 소화기 발사 장치, 제압 모듈 등 주요 하드웨어의 상태를 점검하고 이상 감지 시 즉시 대시보드에 알림을 전송하고 해당 기능을 제한합니다.
- **배터리 부족:** 배터리가 20% 미만일 경우 AMR은 즉각 도킹 스테이션으로 복귀합니다.

---

10. 테스트 및 검증

1. **단위 테스트:** ROS2 노드 및 Flask 서버, YOLO 모델을 포함한 모든 소프트웨어 구성 요소를 개별적으로 테스트하여 기능을 검증합니다.
2. **통합 테스트:** 두 대의 AMR과 모니터링 PC를 통합하여 원활한 데이터 흐름과 명령 실행을 보장합니다.
3. **사용자 승인 테스트(UAT):** 담당자와 함께 대시보드, 경고 및 수동 제어 기능을 실제 상황에서 검증합니다.
4. **현장 테스트:** 목표 보안 구역에서 AMR을 테스트하여 자율 네비게이션, 화재 탐지 및 진압 기능, 직원 대피, 침입자 감시/추적 조건에서 검증합니다.

---

11. 배포 및 유지보수 계획

- **배포 단계:**
    - PC에 ROS2, Flask, SQLite3를 설치합니다.
    - 모니터링 대시보드를 구성하고 AMR을 Wi-Fi로 연결합니다.
    - 초기 네비게이션 경로를 설정하고 경고 기능을 테스트합니다.
- **유지보수 일정:**
    - 매월 AMR과 PC의 소프트웨어 업데이트로 보안 패치 적용
    - 분기마다 AMR의 하드웨어 점검, 센서 보정 및 배터리 검사

---

12. 부록

- **라이브러리 및 종속성:**

    - ROS2 (Humble) - Humble Hawksbill : 로봇 운영체제, 메시지 통신 및 제어

- OpenCV – 4.11.0 : 영상 처리
- Ultralytics YOLOv8 – v8.1 : 객체 탐지 AI 모델
- Flask – 2.3.3 : 로컬 웹 대시보드 서버
- SQLite3 – 3.x : 로컬 데이터 저장
- pyttsx3 – 2.90 : TTS (텍스트 음성 변환) 라이브러리
- RPLIDAR SDK – 1.12.0 : LiDAR 장치용 드라이버

- **용어 사전:**
  - **AMR** (Autonomous Mobile Robot) : 스스로 주행하며 작업을 수행하는 자율 이동 로봇
  - **SLAM** (Simultaneous Localization and Mapping) : 로봇이 자신의 위치를 추정하며 동시에 지도를 만드는 기술
  - **YOLO** (You Only Look Once) : 이미지에서 객체를 실시간으로 탐지하는 딥러닝 알고리즘
  - **ROS2** (Robot Operating System 2) : 로봇 소프트웨어 프레임워크로 노드 간 통신, 센서 데이터 처리 등을 지원
  - **TTS** (Text-to-Speech) : 텍스를 음성으로 변환해주는 기술
  - **LiDAR** (Light Detection and Ranging) : 레이저 기반 거리 측정 장치
  - **Depth Camera** : 거리 정보를 포함한 이미지(깊이 영상)를 제공하는 카메라

- **참고 문헌:**

  - https://www.hellot.net/news/article.html?no=101528 배경 기사 출처
  - ROS2 Documentation – https://docs.ros.org/en/humble/index.html
  - Raspberry Pi Documentation – https://www.raspberrypi.com/documentation/
  - TurtleBot4 Product Page – https://turtlebot.github.io/turtlebot4-user-manual/
  - Ultralytics YOLOv8 GitHub – https://github.com/ultralytics/ultralytics
  - OpenCV Docs – https://docs.opencv.org
  - Create® 3 Robot Documentation (iRobot) – https://edu.irobot.com/what-we-offer/create-3

- pyttsx3 TTS GitHub – https://github.com/nateshmbhat/pyttsx3
- SLAM 설명: "Simultaneous Localization and Mapping: Part I" (IEEE Tutorial)